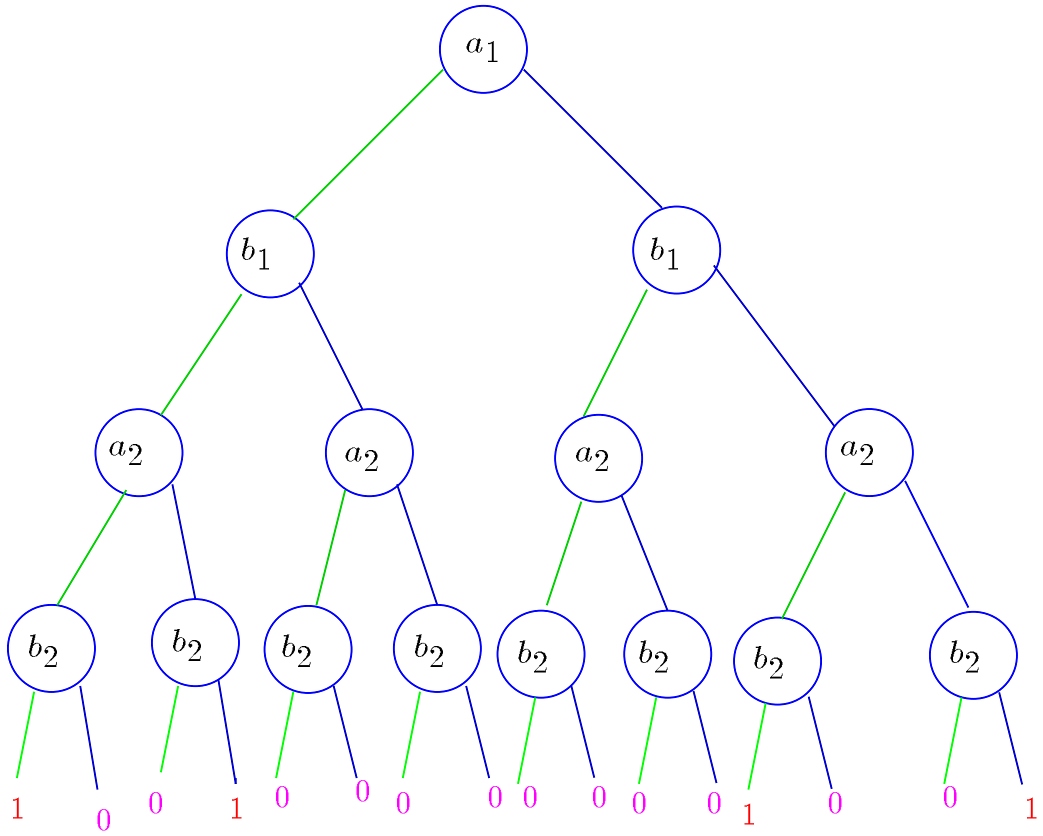
Binary Decision Trees

Consider the Boolean function $f(a_1, a_2, b_1, b_2)$ with

$$f(a_1, a_2, b_1, b_2) = (a_1 \leftrightarrow b_1) \wedge (a_2 \leftrightarrow b_2).$$

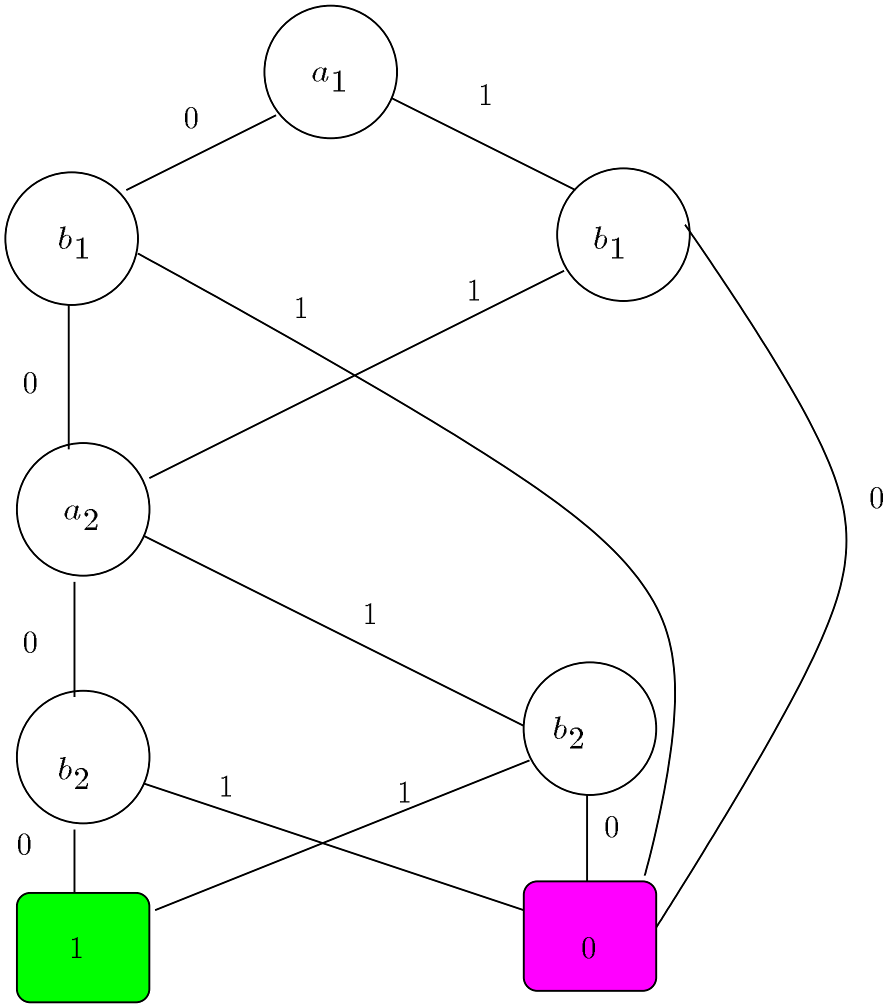


Reduced Ordered Binary Decision Diagram

An ROBDD is obtained from a Binary Decision Diagram by repeatedly applying the following transformations rules:

- Remove duplicate terminals: Eliminate all but one terminal vertex with a given label and redirect all arcs of the eliminated vertices to the remaining one.
- Remove duplicate nonterminals: If two nonterminals are labe-led by the same variable and they are the roots of two isomorphic trees then eliminate of them and redirect all incoming to the remaining one.
- Remove redundant node: If both edges of a node point to the same tree then eliminate this node and redirect all incoming arcs to the pointed tree.

The ROBDD of the two-bit Comparator



Why are ROBDDs better than other Representations?

- Canonicity: For every Boolean function and variable ordering there is a unique ROBDDS representing that function.
- Equality check of Boolean functions in constant time.
- Satisfiability and validity can be solved in constant time.

But:

- ROBDD size depends on the ordering.
- Finding a good ordering is NP-complete.
- There are functions for which for every variable ordering the corresponding ROBDD is exponential in the number of variables (e.g., Multiplication of n-bits).

Bryant's *Apply* Algorithm

Given the ROBDDs of the Boolean functions f and f' and given a Boolean operation $\star$ to compute the ROBDD of $f \star f'$.

Idea is provided by the Shannon expansion:

$$f \star f' = (\neg x \wedge f[0/x] \star f'[0/x]) \vee (x \wedge f[1/x] \star f'[1/x]).$$

Let v and v' be the roots of the ROBDDs of f and f' and let them be marked by x and x' respectively.

- If v and v' are terminals then return $f \star f'$.
- If $x < x'$ then $f'[0/x] = f'[1/x] = f'$ and

  $$f \star f' = (\neg x \wedge f[0/x] \star f') \vee (x \wedge f[1/x] \star f').$$

  The ROBDD of f is computed recursively.
- If $x = x'$ then use the Shannon expansion rule to compute the ROBDD of f recursively.

Remark: Use hash tables to make the algorithm polynomial.